



## PATENT ABSTRACTS OF JAPAN

(11) Publication number: **09081604 A**

(43) Date of publication of application: 28 . 03 . 97

(51) Int. Cl. G06F 17/50

(21) Application number: **07231579**

(22) Date of filing: 08 . 09 . 95

(71) Applicant: **TOSHIBA CORP**

(72) Inventor: AIHARA MASAMI  
FUJITA TORU

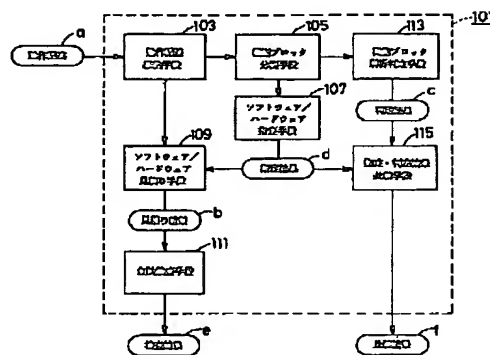
(54) SOFTWARE/HARDWARE COOPERATION  
DESIGN SYSTEM AND DESIGN METHOD  
THEREFOR

(57) Abstract:

**PROBLEM TO BE SOLVED:** To provide a software/hardware cooperation design system and the design method capable of efficiently finding the division of a software and a hardware for satisfying constraints provided beforehand.

**SOLUTION:** Respective plural function blocks for which an operation description for describing the operation of a logic system is divided are analyzed and whether or not it is suitable for realization by the hardware is judged by a function block analysis and judgement means 113. The judged result (c) and the specified result of a software/hardware specifying means 107 are compared and a specified/judged result comparison means 115 clearly demonstrates the function block for which specifying different from the judged result (c) is performed.

COPYRIGHT: (C)1997,JPO



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-81604

(43) 公開日 平成9年(1997)3月28日

(51) Int.Cl.<sup>6</sup>

G 0 6 F 17/50

識別記号

片内整理番号

F I

G 0 6 F 15/60

技術表示箇所

6 0 4 H

6 1 2 C

6 3 6 G

6 5 2 R

審査請求 未請求 請求項の数 2 O L (全 10 頁)

(21) 出願番号 特願平7-231579

(22) 出願日 平成7年(1995)9月8日

(71) 出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72) 発明者 相原 雅己

神奈川県川崎市幸区堀川町580番1号 株式会社東芝半導体システム技術センター内

(72) 発明者 藤田 透

神奈川県川崎市幸区堀川町580番1号 株式会社東芝半導体システム技術センター内

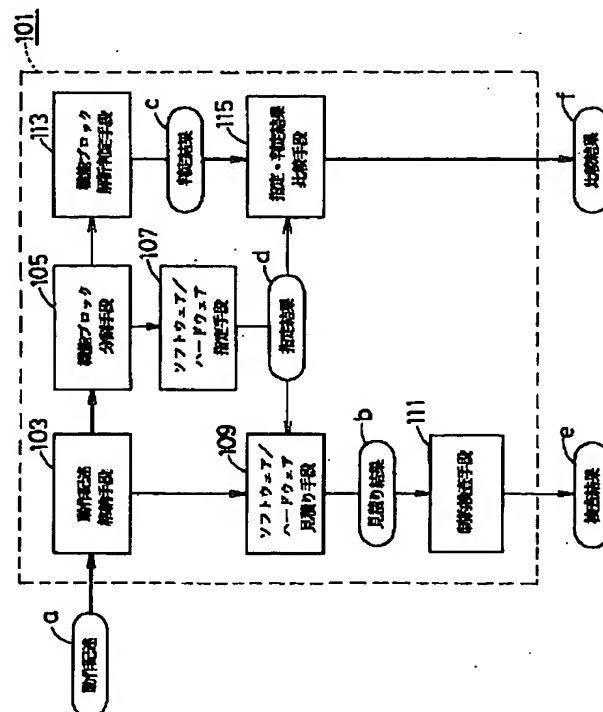
(74) 代理人 弁理士 三好 秀和 (外3名)

(54) 【発明の名称】 ソフトウェア/ハードウェア協調設計システム及びその設計方法

(57) 【要約】

【課題】 本発明は、上記の事情を考慮してなされたもので、その目的とするところは、予め設けられた制約を満たすようなソフトウェアとハードウェアの分割を効率良く見つけることができるソフトウェア/ハードウェア協調設計システム及びその設計方法を提供することを課題とする。

【解決手段】 論理システムの動作を記述した動作記述を分割した複数の機能ブロックそれぞれを解析し、ハードウェアで実現することに適しているか否かを機能ブロック解析判定手段113が判定し、その判定結果cとソフトウェア/ハードウェア指定手段107の指定結果を比較し、判定結果cと異なる指定がされている機能ブロックを指定・判定結果比較手段115が明示するように構成されている。



## 【特許請求の範囲】

【請求項1】 ソフトウェアとハードウェアから構成される論理システムを設計するソフトウェア／ハードウェア協調設計システムであって、

ソフトウェアとハードウェアから構成される論理システムの動作を記述した動作記述を格納する動作記述格納手段と、前記動作記述格納手段に格納された前記動作記述を複数の機能ブロックに分割する機能ブロック分割手段と、前記複数の機能ブロックそれぞれに対し、ソフトウェアで実現するかハードウェアで実現するかを指定するソフトウェア／ハードウェア指定手段と、ソフトウェアによる実現を指定された機能ブロックに対してはプログラムサイズと処理時間を見積り、ハードウェアによる実現を指定された機能ブロックに対してはハードウェア量と処理時間を見積るソフトウェア／ハードウェア見積り手段と、前記ソフトウェア／ハードウェア見積り手段の見積り結果が予め設けられた制約を満たすか否かを検査する制約検査手段とを有するソフトウェア／ハードウェア協調設計システムにおいて、

前記複数の機能ブロックそれぞれの動作記述を解析し、ハードウェアで実現することに適しているか否かを判定する機能ブロック解析判定手段と、

前記機能ブロック解析判定手段の判定結果と前記ソフトウェア／ハードウェア指定手段の指定結果を比較し、前記判定結果と異なる指定がされている機能ブロックを明示する指定・判定結果比較手段とを具備することを特徴とするソフトウェア／ハードウェア協調設計システム。

【請求項2】 ソフトウェアとハードウェアから構成される論理システムの動作を記述した動作記述を格納する第1のステップと、

前記第1のステップで格納された動作記述を複数の機能ブロックに分割する第2のステップと、

前記第2のステップで分割された複数の機能ブロックそれぞれに対し、ソフトウェアで実現するかハードウェアで実現するかを指定する第3のステップと、

前記第3のステップでソフトウェアによる実現を指定された機能ブロックに対してはプログラムサイズと処理時間を見積り、ハードウェアによる実現を指定された機能ブロックに対してはハードウェア量と処理時間を見積る第4のステップと、

前記第4のステップの見積り結果が予め設けられた制約を満たすか否かを検査する第5のステップと、

前記第5のステップにおける検査結果において、前記見積り結果が前記制約を満たさない場合には、前記第2のステップで分割された複数の機能ブロックそれぞれの動作記述を解析し、ハードウェアで実現することが適しているか否かを判定する第6のステップと、

前記第6のステップの判定結果と前記第3のステップの指定結果を比較し、前記判定結果と異なる指定がされている機能ブロックを明示する第7のステップと、 前記

第7のステップの比較結果に基づき前記第3のステップにおける指定を変更し、前記第5のステップにおいて前記第4のステップの見積り結果が予め設けられた制約を満たすまで、前記第3のステップから前記第7のステップを繰り返す第8のステップとを具備することを特徴とするソフトウェア／ハードウェア協調設計方法。

## 【発明の詳細な説明】

## 【0001】

【発明の属する技術分野】本発明は、ソフトウェアとハードウェアから構成される論理システムを設計する際、ソフトウェアとハードウェアを同時に設計することを支援するソフトウェア／ハードウェア協調設計システムに関する。

## 【0002】

【従来の技術】論理システムの実現を行う場合には、ハードウェアにより実現する方法若しくは汎用プロセッサを利用してソフトウェアにより実現する方法が一般的に用いられている。ところが、近年の微細加工技術の発展に伴いLSIが大規模化してきているために、上記2つの方法を併用するケースが多くなってきている。このような場合、従来では、まず最初に設計者の経験等によりハードウェア部分とソフトウェア部分の切り分けを行ない、その後でハードウェアとソフトウェアをそれぞれ独立に設計するという手順で設計が行われていた。

【0003】しかし、この手順では、設計が進展してハードウェアの全体像が具体的にいった段階で、予め設けられた論理システムの制約とのミスマッチが生じる可能性が高く、仮に、このようなミスマッチが生じた場合には、ハードウェアで実現されている機能ブロックをソフトウェアによる実現に変更してハードウェア量を削減する、ソフトウェアで実現されている機能ブロックをハードウェアによる実現に変更して処理の高速化を図る、というように最初に設計者の経験等により行ったソフトウェアとハードウェアの切り分けを変更しなければならなくなり、多くの場合、最初から設計をやり直す必要が生じてしまうのである。このため、設計期間が長くなってしまいう問題が起きていた。

【0004】この問題を解決するため、論理システムの動作機能を記述した動作記述を用いて、ソフトウェアに切り分けされた部分に対してはプログラムサイズと処理時間を、ハードウェアに切り分けされた部分に対してはハードウェア量と処理時間を設計を始める前に見積ることにより、その見積り結果が予め設けられた論理システムの制約を満たすようにソフトウェアとハードウェアの切り分けの変更を繰り返し、制約を満たすことが確認できた段階で初めて実際の設計を開始するというソフトウェア／ハードウェア協調設計と呼ばれる設計手順が提案されている。

【0005】この手順によれば、設計を始める前に、予

10

20

30

40

50

め制約を満たすようソフトウェアで実現する機能ブロックとハードウェアで実現する機能ブロックを分けることができるので、上述した設計のやり直しによる設計時間の増大を解決することができる。

【0006】しかしながら、この手順では、制約を満たすようにソフトウェアとハードウェアの切り分けの変更を行う際に、制約を満たす切り分けが見つかるまで、設計者がいろいろと組合せを変更しなければならないが、例えばハードウェア量を減らしたい場合にどの機能ブロックをハードウェアからソフトウェアに変えたら効果があるのか、あるいは、処理スピードを速くしたい場合にどの機能ブロックをソフトウェアからハードウェアに変えたら効果があるのかということを判断することができないので、設計者は考え得るすべての組合せを逐次的に試していかなければならならず、効率が非常に悪かった。

【0007】

【発明が解決しようとする課題】以上説明したように、設計者の経験に基づいてソフトウェアとハードウェアを切り分けし、その後、ソフトウェアとハードウェアを別々に設計するという、従来の設計手順では、設計がある程度進展した段階になって制約とのミスマッチが生じる可能性が高く、結果として多くの場合は最初から設計をやり直す必要が生じるので、設計期間が長くなるという問題があった。

【0008】一方、ソフトウェア部分及びハードウェア部分それぞれのプログラムサイズ、ハードウェア量、処理時間を見積り、その見積り結果が制約を満たすようなソフトウェアとハードウェアの切り分けを見つけてから、ソフトウェアとハードウェアそれぞれの設計を開始するというソフトウェア／ハードウェア協調設計手順では、上記従来の手順に比べて設計がある程度進んだ段階で設計のやり直しをすることがなくなるので設計期間を短縮する効果はあるが、ソフトウェアとハードウェアとの切り分けを決める際に、考え得る組合せすべてを逐次的に試さなければならないため切り分けの決定に時間がかかり、設計期間の点で効率が悪いという問題が生じていた。

【0009】そこで、本発明は、上記の事情を考慮してなされたもので、その目的とするところは、予め設けられた制約を満たすようなソフトウェアとハードウェアの分割を効率良く見つけることができるソフトウェア／ハードウェア協調設計システム及びその設計方法を提供することにある。

【0010】

【課題を解決するための手段】上記目的を達成するため本発明は、ソフトウェアとハードウェアから構成される論理システムを設計するソフトウェア／ハードウェア協調設計システムであって、ソフトウェアとハードウェアから構成される論理システムの動作を記述した動作記述を格納する動作記述格納手段と、前記動作記述格納手段

に格納された前記動作記述を複数の機能ブロックに分割する機能ブロック分割手段と、前記複数の機能ブロックそれぞれに対し、ソフトウェアで実現するかハードウェアで実現するかを指定するソフトウェア／ハードウェア指定手段と、ソフトウェアによる実現を指定された機能ブロックに対してはプログラムサイズと処理時間を見積り、ハードウェアによる実現を指定された機能ブロックに対してはハードウェア量と処理時間を見積るソフトウェア／ハードウェア見積り手段と、前記ソフトウェア／ハードウェア見積り手段の見積り結果が予め設けられた制約を満たすか否かを検査する制約検査手段とを有するソフトウェア／ハードウェア協調設計システムにおいて、前記複数の機能ブロックそれぞれの動作記述を解析し、ハードウェアで実現することに適しているか否かを判定する機能ブロック解析判定手段と、前記機能ブロック解析判定手段の判定結果と前記ソフトウェア／ハードウェア指定手段の指定結果を比較し、前記判定結果と異なる指定がされている機能ブロックを明示する指定・判定結果比較手段とを具備することを特徴とする。

【0011】なお、前記機能ブロック解析判定手段は、各機能ブロックの動作記述が、パイプライン処理を行うハードウェアに適しているか否か、及び、並列処理を行うハードウェアに適しているか否かにより判定することができる。

【0012】

【発明の実施の形態】以下、図面を用いて本発明の実施の形態を説明する。

【0013】図1は、本発明の実施形態に係るソフトウェア／ハードウェア協調設計システムの一構成例を示す図である。

【0014】図1に示すソフトウェア／ハードウェア協調設計システム101は、ソフトウェアとハードウェアから構成される論理システムの動作を記述した動作記述aを格納する動作記述格納手段103と、動作記述格納手段103に格納された動作記述aを複数の機能ブロックに分割する機能ブロック分割手段105と、前記複数の機能ブロックそれぞれに対し、ソフトウェアで実現するかハードウェアで実現するかを指定するソフトウェア／ハードウェア指定手段107と、ソフトウェアによる実現を指定された機能ブロックに対してはプログラムサイズと処理時間を見積り、ハードウェアによる実現を指定された機能ブロックに対してはハードウェア量と処理時間を見積るソフトウェア／ハードウェア見積り手段109と、ソフトウェア／ハードウェア見積り手段109の見積り結果bが予め設けられた制約を満たすか否かを検査し、検査結果eを出力する制約検査手段111と、前記複数の機能ブロックそれぞれの動作記述aを解析し、ハードウェアで実現することに適しているか否かを判定する機能ブロック解析判定手段113と、機能ブロック解析判定手段113の判定結果cとソフトウェア／

ハードウェア指定手段107の指定結果dを比較し、判定結果cと異なる指定がされている機能ブロックを明示することで比較結果fを出力する指定・判定結果比較手段115から構成されている。

【0015】次に、動作記述aの具体例を用いて本発明の実施形態に係るソフトウェア／ハードウェア協調設計システムの動作を図面を参照しながら説明する。なお、図2は、図1に示す動作記述aの一例を示す図であり、機能ブロック201、202、203から構成されている。また、図3は、本発明の実施形態に係るソフトウェア／ハードウェア協調設計の手順を示すフローチャートである。

【0016】図3において、まず、動作記述格納手段103は、動作記述aを格納する（ステップ1）。

【0017】次に、機能ブロック分割手段105は、ステップ1で動作記述格納手段103に格納された動作記述aを機能ブロック201、202、203に分割する（ステップ2）。

【0018】次に、ソフトウェア／ハードウェア指定手段107は、ステップ2で動作記述aを分割した機能ブロック201、202、203それぞれに対して、ソフトウェアで実現するかハードウェアで実現するかを指定する。図4は、機能ブロック201、202、203それぞれの実現方法についての指定結果の一例を示す図である。ここでは、機能ブロック201と機能ブロック202にはソフトウェアによる実現、機能ブロック203にはハードウェアによる実現をそれぞれ指定している（ステップ3）。

【0019】次に、ソフトウェア／ハードウェア見積り手段109は、ステップ3でソフトウェアによる実現を指定された機能ブロック201及び202に対してはプログラムサイズと処理時間の見積りを行い、一方、ハードウェアによる実現を指定された機能ブロック203に対してはハードウェア量と処理時間の見積りを行う（ステップ4）。

【0020】次に、制約検査手段111は、ステップ4でソフトウェア／ハードウェア見積り手段109が見積った見積り結果が予め設けられた制約を満たすか否かを検査し、見積り結果が制約を満たす場合には、ステップ3での指定結果に基づきソフトウェアとハードウェアをそれぞれ作成し、所望の論理システムの設計はここで終了する。（ステップ5）。

【0021】次に、ステップ5において見積り結果が制約を満たさない場合、機能ブロック解析判定手段113は、ステップ2で分割された機能ブロック201、202、203それぞれについて、その動作記述を解析し、ハードウェアに適した記述か否かを判定する。図5は、機能ブロック201、202、203それぞれについて、ハードウェアによる実現が適しているか否かの判定結果の一例を示す図である（ステップ6）。なお、この

判定方法については後で説明する。

【0022】次に、指定・判定結果比較手段115は、ステップ3でソフトウェア／ハードウェア指定手段107が行った指定結果（図4）とステップ6で機能ブロック解析判定手段113が行った判定結果（図5）を比較し、判定結果と異なる指定がされている機能ブロックを明示する。図6は、図4に示す指定結果と図5に示す判定結果の比較結果の一例を示す図である（ステップ7）。

【0023】最後に、図6に示す比較結果に基づいて、ステップ2における指定を変更し、ステップ5においてステップ4における見積り結果が制約を満たすまでステップ3からステップ7を繰り返す（ステップ8）。なお、この変更方法についても後で説明する。

【0024】次に、ステップ6の判定方法について説明する。

【0025】図2に示す機能ブロック201については、まず、変数の参照関係を解析する。値を参照している変数は配列data、aa、x、yであり、値を設定している変数は配列illu、aa、x、yであることがわかる。この内、x、yは配列のインデックスとしてのみ使用され、aaは設定／参照の順序関係から一時的に値を保持するために使用されていることもわかる。これにより配列dataを機能ブロック201の入力、配列illuを出力とみなす。

【0026】次に、入力と出力の依存関係からハードウェアに適した記述か否かを判定する。1回のループでdata[x][y]、data[x][y-1]、data[x-1][y]の3つの値を使って計算しており、また規則的であることから、パイプライン処理を行なうハードウェアに適した記述であると判定する。また、入力のみから出力を計算しておりループ1回毎の計算はそれぞれ独立していることから、並列処理を行なうハードウェアに適した記述であると判定する。

【0027】上述したように、機能ブロック201は並列処理またはパイプライン処理を行なうハードウェアに適したものであり、この時、図5に示すように、判定結果は“適している”と表示される。

【0028】図2に示す機能ブロック202については、変数の参照関係を解析した結果、配列illuを機能ブロックの入力、配列viを出力、変数dictはループ内部で参照が先にあり、設定が後にあることから前のループの計算結果を次のループの計算で使うために一時的に保持している変数とみなす。1回のループでillu[m][n]の値を使って計算しているが、この場合、m、nの値の決め方に規則性がないためパイプライン処理を行なうハードウェアに適した記述ではないと判定する。また、変数dictにより1回前のループでの計算結果を使用するため、1回毎の計算に依存関係が生じるため並列処理を行なうハードウェアに適した記述で

はないと判定する。従って、機能ブロック202はハードウェアに適した記述ではないと判定する。

【0029】 上述したように、機能ブロック202は並列処理及びパイプライン処理を行うハードウェアに適したものではないので、図5に示すように、判定結果は“適していない”と表示される。

【0030】 図2に示す機能ブロック203についても同様に、変数の参照関係を解析した結果、配列*v i*を機能ブロック203の入力、配列*v o*を出力、変数*x d*、*y d*もループ内部で参照が先にあり設定が後にあるため10 前のループの計算結果を次のループでの計算に使うために一時的に保持している変数とみなす。1回のループで*v i* [*x*]、*v i* [*x*-1] の値を使って計算しており規則てきであることからパイプライン処理を行なうハードウェアに適した記述であると判定する。しかし、変数*x d*、*y d*により1回前のループでの計算結果を使用するため、1回毎の計算に依存関係が生じるため並列処理を行なうハードウェアに適した記述ではないと判定する。

【0031】 上述したように、機能ブロック203はパイプライン処理を行うハードウェアに適したものであり、図5に示すように判定結果は“適している”と表示される。

【0032】 次に、ステップ8の指定変更方法について説明する。

【0033】 図6において、指定・判定結果比較手段115は、ステップ6でハードウェアによる実現が適していると判定されているにもかかわらず、ステップ2でソフトウェアによる実現が指定されている機能ブロック201をハイライト表示により明示している。これにより、設計者は、制約を満たすようにソフトウェアで実現する機能ブロックとハードウェアで実現する機能ブロックの指定の変更を行う。

【0034】 例えば、図4に示すソフトウェア／ハードウェア指定では処理時間が長すぎるという理由から制約を満たさない場合、ソフトウェアによる実現を指定されている機能ブロック201及び203のうち、ハードウェアによる実現が適している機能ブロックをソフトウェアによる実現からハードウェアによる実現に変更すれば、処理時間を短縮することができる。従来では、設計者がこの2つの機能ブロックについて指定をどう変えれば制約が満たされるかを試行錯誤的に試さなければならなかったが、本発明では図6に示す表示を見れば、制約を満たすソフトウェア／ハードウェアの指定を効率良く見つけることが可能である。

【0035】 従って、ハイライト表示されている機能ブロック201の指定を変えることで処理時間を短縮することができるのである。なお、この場合、ハードウェア量の制約を考慮して並列処理あるいはパイプライン処理を選択すれば良い。

【0036】 図7は、図2に示す機能ブロック201、202、203それぞれの実現方法についての指定結果の他の一例を示す図である。ここでは、機能ブロック201、202及び203はハードウェアによる実現をそれぞれ指定している。

【0037】 図8は、図7に示す指定結果と図5に示す判定結果を指定・判定結果比較手段115により比較し、判定結果と異なる指定をされている機能ブロックを明示した一例を示す図であり、機能ブロック202がハードウェアに適した記述でないにもかかわらずハードウェアによる実現が指定されているためハイライト表示されている。

【0038】 例えば、図7に示すソフトウェア／ハードウェア指定ではハードウェア量が多すぎて制約を満たさないとする。この場合、ハードウェアによる実現を指定されている機能ブロックをソフトウェアによる実現に変えることで、ハードウェア量を減らすことができるので、図7に示す表示を見て、ハイライト表示されている機能ブロック202をハードウェアによる実現からソフトウェアによる実現に変えることで、処理時間にあまり影響を与えないでハードウェア量を減らすことができる。

【0039】 ここで、図2に示す機能ブロックのハードウェアによる実現の一例を示しておく。

【0040】 図9及び図10は図2に示す機能ブロック201をハードウェアで実現した場合の論理回路の一例を示す図であり、図9は並列処理を行なうハードウェア、図10はパイプライン処理を行なうハードウェアを示している。また、図11は図2に示す機能ブロック202をハードウェアで実現した場合の論理回路の一例を示す図、図12は図2に示す機能ブロック203をハードウェアで実現した場合の論理回路の一例を示す図である。

【0041】 図9～図12から明らかなように、図10及び図12のパイプライン処理する論理回路のハードウェア量は少ないが、図9の並列処理する論理回路及び図11のハードウェアに適した記述でない機能ブロックの論理回路はハードウェア量が多いことがわかる。

【0042】 一方、ソフトウェアで実現した機能ブロックは、プロセッサとメモリが必要であるが、他のブロックと共用可能であるために無視するとハードウェア量はゼロとみなせる。また、一般的に、ソフトウェアよりもハードウェアが、逐次処理よりもパイプライン処理のほうが、パイプライン処理よりも並列処理のほうが、処理時間が短くなる傾向がある。

【0043】 従って、機能ブロック201及び機能ブロック203の動作はパイプライン処理あるいは並列処理を行うハードウェアで実現することにより処理時間を大幅に短縮できる。一方、機能ブロック202の動作は逐次的であり、ハードウェアで実現しても処理時間を大幅

に短縮することはできず、また、ハードウェア量も多いので、ソフトウェアによる実現が適しているのである。

#### 【0044】

【発明の効果】以上詳述したように本発明によれば、論理システムの動作を記述した動作記述を解析して機能ブロックそれぞれに対してハードウェアに適した記述か否かを判定し、設計者が指定した結果と比較することにより、ハードウェアに適した記述にも拘らずソフトウェアによる実現を指定されている機能ブロックあるいはハードウェアに適していない記述にも拘らずハードウェアによる実現を指定されている機能ブロックを明示することができる。これにより、制約が満たされず指定を変更する場合に、どの機能ブロックの指定を変えたらハードウェア量を効果的に削減することができるかあるいは処理時間を効果的に短縮できるかが容易に判定できるため、制約を満たすようなソフトウェアとハードウェアの指定を無駄な試行錯誤をすることなく求めることが可能になり、結果的に設計期間が短縮されるという効果がある。

#### 【図面の簡単な説明】

【図1】本発明の実施形態に係るソフトウェア／ハードウェア協調設計システムの一構成を示す図である。

【図2】図1に示す動作記述aの一例を示す図である。

【図3】本発明の実施形態に係るソフトウェア／ハードウェア協調設計の手順を示すフローチャートである。

【図4】図2に示す機能ブロック201、202、203それぞれの実現方法についての指定結果の一例を示す図である。

【図5】図2に示す機能ブロック201、202、203それぞれについて、ハードウェアによる実現が適しているか否かの判定結果の一例を示す図である。

\* 30

\* 【図6】図4に示す指定結果と図5に示す判定結果の比較結果の一例を示す図である。

【図7】図2に示す機能ブロック201、202、203それぞれの実現方法についての指定結果の他の一例を示す図である。

【図8】図7に示す指定結果と図5に示す判定結果を指定・判定結果比較手段115により比較し、判定結果と異なる指定をされている機能ブロックを明示した一例を示す図である

10 【図9】図2に示す機能ブロック201を並列処理を行うハードウェアで実現した場合の論理回路の一例を示す図である。

【図10】図2に示す機能ブロック201をパイプライン処理を行うハードウェアで実現した場合の論理回路の一例を示す図である。

【図11】図2に示す機能ブロック202をハードウェアで実現した場合の論理回路の一例を示す図である。

【図12】図2に示す機能ブロック203をハードウェアで実現した場合の論理回路の一例を示す図である。

#### 【符号の説明】

- 101 ソフトウェア／ハードウェア協調設計システム
- 103 動作記述格納手段
- 105 機能ブロック分割手段
- 107 ソフトウェア／ハードウェア指定手段
- 109 ソフトウェア／ハードウェア見積り手段
- 111 制約検査手段
- 113 機能ブロック解析判定手段
- 115 指定・判定結果比較手段
- 201、202、203 機能ブロック

【図4】

機能ブロック	実現方法
201	ソフトウェア
202	ソフトウェア
203	ハードウェア

【図5】

機能ブロック	並列処理	パイプライン	判定結果
201	○	○	適している
202	×	×	適していない
203	×	○	適している

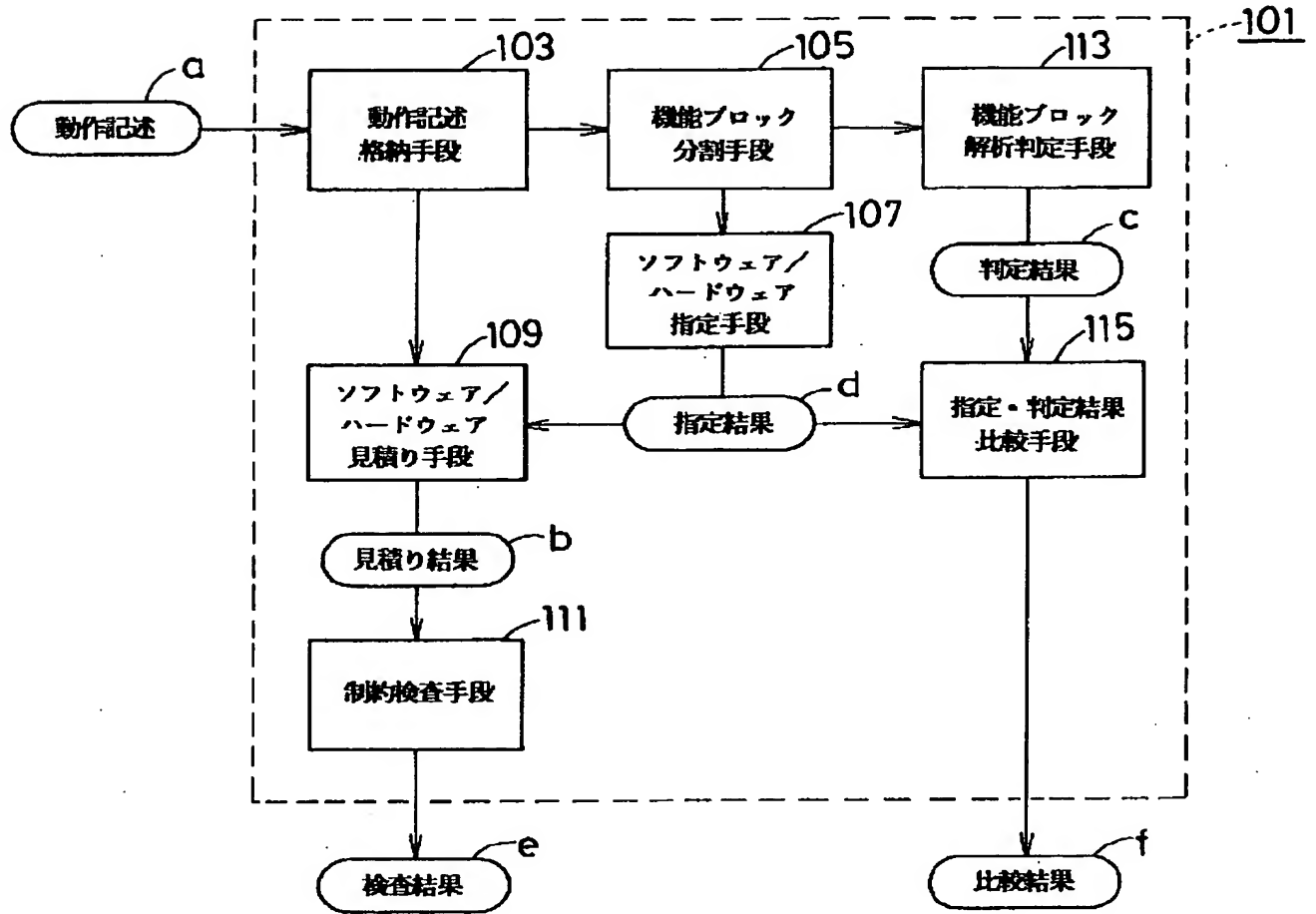
【図7】

機能ブロック	実現方法
201	ハードウェア
202	ハードウェア
203	ハードウェア

【図8】

機能ブロック	実現方法	並列処理	パイプライン	判定結果
201	ハードウェア	○	○	適している
202	ソフトウェア	×	×	適していない
203	ハードウェア	×	○	適している

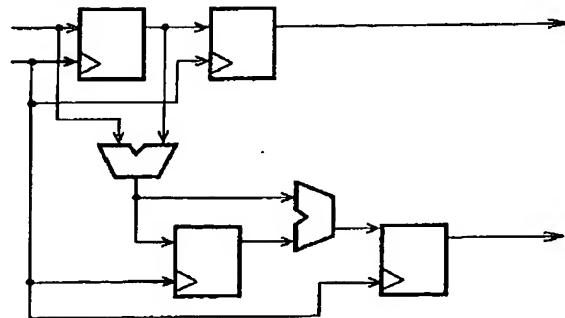
【図 1】



【図 6】

機能ブロック	実現方法	並列処理	パイプライン	判定結果
201	ハードウェア	○	○	適している
202	ソフトウェア	×	×	適していない
203	ハードウェア	×	○	適している

【図 12】





【図2】

201

```

for(int y=1;y<ymax;y++) {
    for(int x=1;x<xmax;x++) {
        if(abs(aa=(data[x][y]-data[x-1][y]))>=1)
            if (aa>0)
                illu[x][y] |= 1;
            else
                illu[x-1][y] |= 1;
        else
            illu[x][y]=0;
        if(abs(aa=(data[x][y]-data[x][y-1]))>=1)
            if (aa>0)
                illu[x][y] |= 1;
            else
                illu[x][y-1] |= 1;
    }
}

```

202

```

for(int y_st=1;y_st<ymax;y_st++) {
    for(int x_st=1;x_st<xmax;x_st++) {
        if (illu[x_st][y_st]) {
            int x=x_st, y=y_st, dict=0;
            while(1) {
                v[vidx].x=x;
                v[vidx++].y=y;
                v[vidx++].fig=0;
                dict=get_next(&x,&y,dict);
                if ((x==x_st)&&(y==y_st)) goto Block3;
            }
        }
    }
}

```

{Block3:

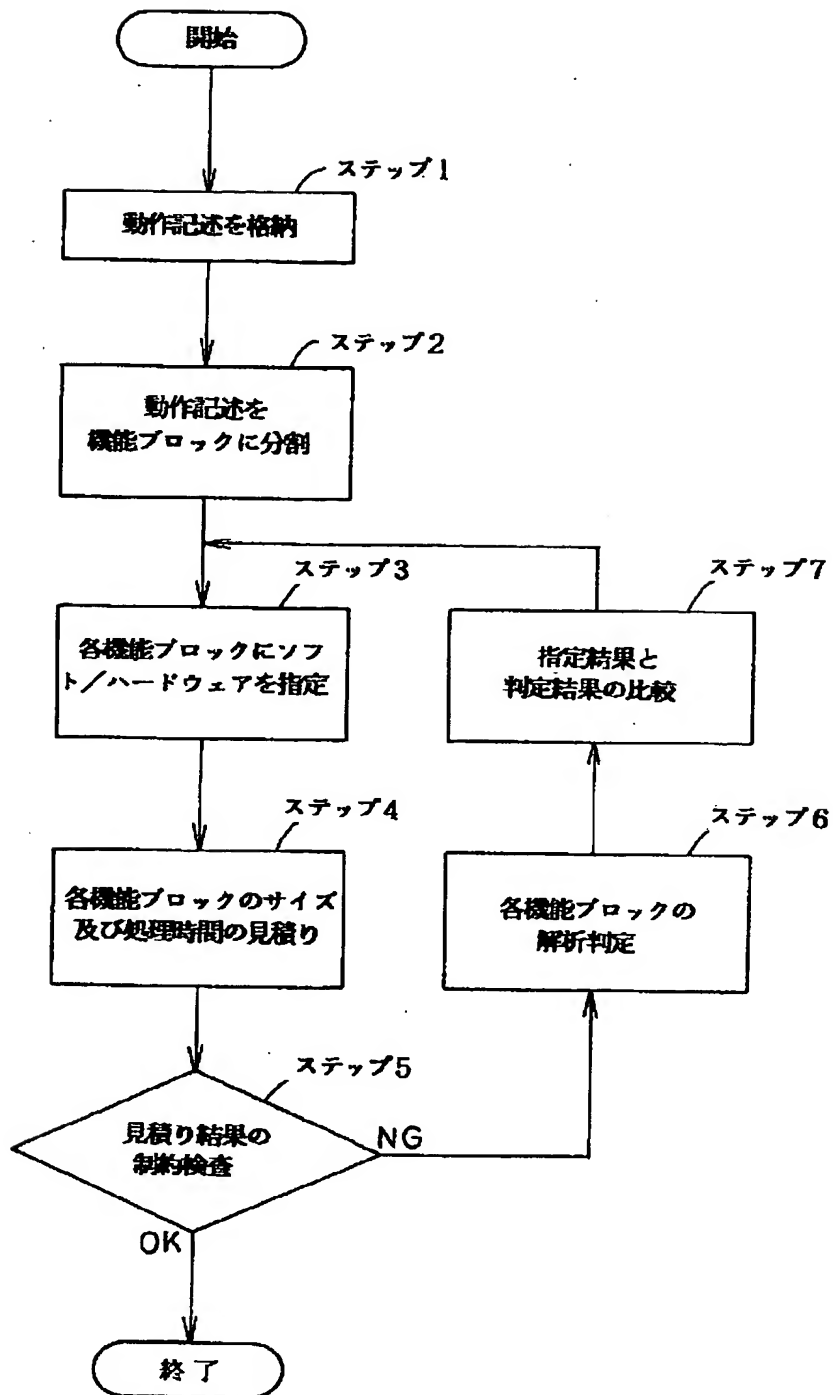
```

for(int i=1;i<vidx;i++) {
    if(xd!=v[i].x-v[i-1].x) {
        xd=v[i].x-v[i-1].x;
        vo[i].fig|=1;
    }
    if(yd!=v[i].y-v[i-1].y) {
        yd=v[i].y-v[i-1].y;
        vo[i].fig|=1;
    }
    vo[i].x=v[i].x;
    vo[i].y=v[i].y;
}

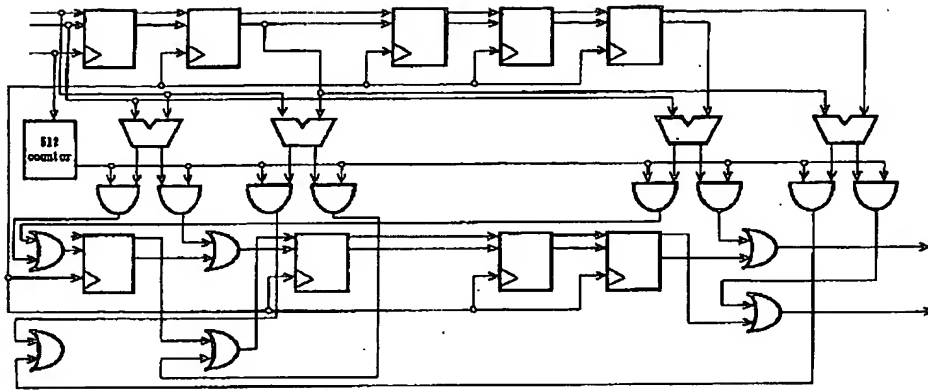
```

203

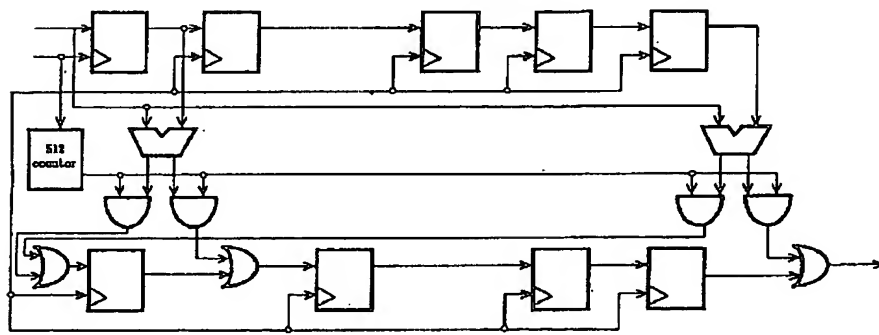
【図3】



【図9】



【図10】



【図11】

